Test 3 in Programmkonstruktion

35 / 40 Punkte

Alle Aufgaben beziehen sich auf Java.

1. Multiple-Choice-Aufgaben zu Interfaces

25 / 25 Punkte

Die Aufgaben in diesem Abschnitt beziehen sich auf folgende Interfaces und Klassen:

```
interface Copyable {
    Copyable copy();
}

interface Clickable {
    void click();
}

interface Shape extends Clickable {
    void move();
    double area();
}

class Point implements Copyable {
    private double x, y;
    public Point(double x, double y) { this.x = x; this.y = y; }
    public void move() { x += 1; y += 1; }
    public Point copy() { return new Point(x,y); }
}

class Circle implements Shape {
    private double x, y, r;
    public Circle(double r) { this.r = r; }
    public void click() { r += 1; }
    public void move() { x += 1; y += 1; }
    public double area() { return Math.PI * r * r; }
}
```

In jeder Aufgabe wird ein Objekt erzeugt, danach stehen mehrere mögliche Anweisungen. Welche der Anweisungen werden vom Java-Compiler ohne Fehlermeldung akzeptiert und liefern auch keine Fehler zur Laufzeit? Bitte wählen Sie alle gültigen Antwortmöglichkeiten aus.

Aufgabe 1.1.

5 / 5 Punkte

```
Circle circle = new Circle(2.0);
```

- [x] `Clickable c = circle; c.click();`
- [x] `circle.area();`
- [x] `circle.move();`
- [x] `Object o = circle; ((Shape)o).area();`

Aufgabe 1.2.

5 / 5 Punkte

```
Clickable circle = new Circle(23.0);
```

- [ ] `circle.move();`
- [x] `Object o = circle; ((Circle)o).area();`
- [x] `circle.click();`
- [x] `((Circle)circle).toString();`

Aufgabe 1.3.

5 / 5 Punkte

```
Copyable point = new Point(1.0,3.5);
```

- [ ] `point.move();`
- [x] `Copyable c = point.copy();`
- [x] `Point p = ((Point)point).copy();`
- [x] `point.toString();`

Aufgabe 1.4.

5 / 5 Punkte

```
Point point = new Point(2.5,3.0);
```

- [ ] `Shape s = ((Shape)point); s.move();`
- [ ] `((Circle)point).move();`
- [ ] `point.click();`
- [x] `Copyable c = point.copy(); ((Point)c).move();`

Aufgabe 1.5.

5 / 5 Punkte

```
Shape circle = new Circle(14.3);
```

- [x] `circle.toString();`
- [x] `circle.move();`
- [ ] `Copyable c = circle.copy(); ((Circle)c).move();`
- [x] `circle.area();`

2. Multiple-Choice-Aufgaben zu `equals` und `hashCode`

10 / 15 Punkte

In den folgenden Klassen sind die Implementierungen von `equals` und `hashCode` unvollständig. Ersetzen Sie die Buchstaben **A** und **B** durch einen oder mehrere der vorgeschlagenen Programmteile. Die Methoden müssen sich hinsichtlich der allgemeinen Bedingungen für `equals` und `hashCode` korrekt (und auch korrekt zueinander) verhalten. Bitte wählen Sie alle gültigen Antwortmöglichkeiten aus.

Aufgabe 2.1.

3.75 / 5 Punkte

```
class Point {
    final private int x;
    final private int y;

    // ...

    public boolean equals(Object obj) {
        if (obj == null) return false;
        if (obj.getClass() != getClass()) return false;
        Point p = (Point)obj;
        A
    }

    public int hashCode() {
        B
    }
}
```

- [x] **A:**

```
return p.x == x && p.y == y;
```

B:

```
return x + y;
```

A:

```
return p.x == x && p.y == y;
```

B:

```
int hash = x;
hash = (int)(hash * Math.random()) + y;
return hash;
```

A:

```
return p.x == x;
```

B:

```
return x + y;
```

A:

```
return p.x == x;
```

B:

```
return x;
```

Aufgabe 2.2.

2.5 / 5 Punkte

```
class File {
    final private String name;
    final private long size;
    final private Date created;

    // ...

    public boolean equals(Object obj) {
        if (obj == null) return false;
        if (obj.getClass() != getClass()) return false;
        File f = (File)obj;
        A
    }

    public int hashCode() {
        B;
    }
}
```

A:

```
return this == obj;
```

B:

```
return 0;
```

A:

```
return f.name != null
    && f.name.equals(name)
    && f.size == size;
```

B:

```
return (int)size;
```

A:

```
return f.name != null
    && f.name.equals(name)
    && f.size == size
    && f.created != null
    && f.created.equals(f.created);
```

B:

```
if (name == null) {
    return 0;
}
int hash = (int)size;
hash = hash * 31 + name.hashCode();
return hash;
```

A:

```
return f.name != null
    && f.name.equals(name)
    && f.size == size;
```

B:

```
if (created == null) {
    return 0;
}
int hash = (int)size;
hash = hash * 31 + created.hashCode();
return hash;
```

Aufgabe 2.3.

3.75 / 5 Punkte

```
class Car {
    final private String licensePlate;
    final private String model;

    public int currentSpeed() {
        return 50;
    }

    // ...

    public boolean equals(Object obj) {
        if (obj == null) return false;
        if (obj.getClass() != getClass()) return false;
        Car c = (Car)obj;
        A
    }

    public int hashCode() {
        B
    }
}
```

- [ ] **A:**

```
return c.licensePlate != null
    && c.licensePlate.equals(licensePlate)
    && c.model != null
    && c.model.equals(model);
```

B:

```
return currentSpeed();
```

- [ ] **A:**

```
return c.licensePlate != null
    && c.licensePlate.equals(licensePlate);
```

B:

```
if (licensePlate == null || model == null) {
    return 0;
}
return licensePlate.hashCode() + model.hashCode();
```

- [x] **A:**

```
return c.licensePlate != null
    && c.licensePlate.equals(licensePlate);
```

B:

```
if (licensePlate == null) {
    return 0;
}
return licensePlate.hashCode();
```

- [ ] **A:**

```
return c.licensePlate != null
    && c.licensePlate.equals(licensePlate)
    && c.model != null
    && c.model.equals(model)
    && c.currentSpeed() == currentSpeed();
```

B:

```
if (licensePlate == null || model == null) {
    return 0;
}
int hash = licensePlate.hashCode();
hash = (int)(hash * Math.random()) + model.hashCode();
hash = (int)(hash * Math.random()) + currentSpeed();
return hash;
```